K I S M E T V 6.0.3

SCRIPT MANUAL

U. Kühnapfel, H.G. Krumm, C. Kuhn, M. Hübner, H. Maaß, K. Çakmak

Forschungszentrum Karlsruhe GmbH
Institut für Angewandte Informatik

Juli 1999

Table of contents

1 The KISMET SCRIPT-Command Interface

1.1 Overview

SCRIPT commands provide a high-level *textual* interface to the **KISMET** command processor.

SCRIPT commands can be entered into KISMET through three different means of input channels:

1. A sequence of SCRIPT-Commands is written into a ***SCRIPT-File***. This file is loaded and executed by KISMET during a simulation session using the popup-menu command *"Exec_SCRIPT-File"*. Optionally, a SCRIPT-filename can be added to the simulation file as the last parameter. This SCRIPT-file will then be executed immediately after model initialization. Therefore, such a SCRIPT-file is called ***Startup SCRIPT***. Typically, the startup SCRIPT is used to configure KISMET with model specific parameters.
2. A single SCRIPT-Command is sent to KISMET from another process via memory buffer using the **UNIX message** facility.
3. A SCRIPT-Command is embedded in the **Comment-Statement** of an **IRDATA-program**. The IRDATA interpreter in KISMET detects SCRIPT-Commands if the IRDATA comment starts with the special character **" % "**.

SCRIPT-Files
Typical applications of SCRIPT-files are for example automatic execution of complex animation sequences which involve several robot programs, changes of view parameters or detailing levels. Another possibility is to build reconfiguration files through SCRIPT files to modify the current model data-tree of detail- and assembly levels (examine the SCRIPT-commands ACTIVATE, DEACTIVATE, SWAP_NODE, LEVEL_UP, LEVEL_DOWN etc.).

The following conventions apply to **SCRIPT-Files**:

- SCRIPT-Files are read by **KISMET** from the directory **'$kis_home/scripts'**. The Shell-Variable 'kis_home' can be set via the C-Shell command **setenv** according to the current project directory. Usually this is done via adding the command line

  ```
  setenv kis_home /usr/people/kis_applikations/project_id
  ```

  into the users login-File **'.cshrc'**.
- SCRIPT-Files are generated **textually**.
- Every command in a SCRIPT-File is **delimited** with the special character **" ; "**. But the 'semicolon' must not be used if the SCRIPT command is sent through the 'message' interface. In the second case is for every SCRIPT command one 'message' to be sent with all required parameters.
- **Comment-Lines** are entered into a SCRIPT-File with the character **'#'** in the first column of the line.

SCRIPT-commands using UNIX-messages
UNIX-'messages' is one possibility to accomplish communications between concurrently and in parallel running processes (programs). Other types of UNIX Inter-Process Communication (IPC)

are 'semaphores', 'pipes', 'shared memory' and of cause the use of common physical data-files.

The message type of IPC allows processes (executing programs) to communicate through the exchange of data stored in buffers. This data is transmitted between processes in discrete portions called 'messages'.

In the case of SCRIPT-commands it is possible, to initiate a KISMET command from another process which is executed in parallel to KISMET. One possibility to use this feature is for example to adapt KISMET to a specific monitoring RH-application without modifying the KISMET kernel itself. The application specific UNIX-process (program) provides the link between KISMET and the robot controllers or between KISMET and a high-level task planning module.

For details about 'message'-programming see the *"IRIS-4D Programmers Guide Vol.II: Chap.8, Interprocess Communication"*.

- KISMET is using the **message-key** '2351' with **message-type** '5' for SCRIPT commands.

1.2 The SCRIPT-Command Language

The different SCRIPT commands can be classified in three functional groups:

1. **General SCRIPT-Commands** which can be used to control a simulation or sequence, and
2. SCRIPT-Commands for definition of **physical ROBOT parameters** or for specification of the simulated **ROBOT controller features**.
3. SCRIPT-Commands for **Animation**. This group of commands controls the viewing and display parameters and can be used together with the other two groups during execution of animation sequences.

The following chapters document the various SCRIPT-Commands (implementation level March 1999).

1.2.1 General SCRIPT-Commands

Comment lines in a SCRIPT-file are those lines with the character '#' in the first position of a line. The whole SCRIPT-line is interpreted as a comment line.

Example:

```
# This is an example of a comment line
```

- **ACTIVATE <frame_node>**

  activates (makes visible) a model-branch (COMPONENT) which was already loaded, or it loads an ABSTRACT-File if the component is not available in the main memory. A **model-branch** is here and in the following parts the equivalent of an '.mpc'-File. The parameter <frame_node> is a textstring which is for KISMET a unique identification of the data node, i.e. a full qualified name. This textstring is composed of the names of FRAMES or JOINTS, i.e. <joint_names> or <frame_name>, respectively.

  The tree-hierarchies or detailing-levels are separated by '.' (dot).

Example:

```
ACTIVATE  Testlab_1.FL_Klein.Flansch_Typ1;
```

- **DEACTIVATE <frame_node>**

  deactivates a model-branch. The model-branch is **not erased** from the KISMET runtime display-list, but the specified part is not scanned for model display or collision test calculations.

  Example:

  ```
  DEACTIVATE  Testlab_1.FL_Klein.Flansch_Typ1;
  ```

- **DELETE_NODE <frame_node>**

  the model-branch is erased from the KISMET runtime data-structure (of cause not from the database). The operation is used to make system memory available for effective rendering of other components which are to be loaded subsequently.

  Example:

  ```
  DELETE_NODE  Testlab_1.FL_Klein.Flansch_Typ1;
  ```

- **SWAP_NODE <frame_node>**

  the more detailed geometry of the specified <frame_node> is replaced for rendering by the less detailed geometry of the father-node or vice versa.

  Example:

  ```
  SWAP_NODE  Testlab_1.FL_Klein.Flansch_Typ1;
  ```

- **LEVEL_DOWN <frame_node>**

  loads and activates down from the specified data-node another layer of 'details', i.e. of '.mpc'-Files. The geometry of the 'father-node' is automatically swapped. The detailing level is increased.

  Example:

  ```
  LEVEL_DOWN  Testlab_1.FL_Klein;
  ```

- **LEVEL_UP <frame_node>**

  deactivates from the specified <frame_node> downwards the next layer of details, i.e. of loaded '.mpc'-Files. The detailing level is decreased.

  Example:

  ```
  LEVEL_UP  Testlab_1.FL_Klein;
  ```

- **ROBOT_LANGUAGE <language>**

  sets the current ROBOT language. All subsequently loaded ROBOT programs are scanned

and parsed according to the specified semantics- and syntax rules. The following set of robot languages or codes can be used:

<language> ::= **irdata** | **srcl** | **jet_boom** | **jet_tarm**

The default robot code is 'irdata'. Example:

```
ROBOT_LANGUAGE  jet_tarm ;
```

- **ROBACT <Robot_id>**

  activates the specified ROBOT as the actual kinematical object, the <Robot_id> is the name of the mechanism as defined in the '.mpc'-file.

  This means more specific, that the input focus for movement commands, TCP definitions etc. is linked to the specified ROBOT. Also for output of joint-variables, TCP-coordinates, ROBOT program step number, execution times etc. the variables of the current ROBOT are used.

  Example:

```
ROBACT PUMA_762 ;
```

- **TF_LOAD <teach_filename> <Robot_id>**

  loads a ROBOT program into the virtual controller of the specified ROBOT. The argument <Robot_id> is the parameter <ROB_name> as defined in the '.mpc'-File of the ROBOT. See the specification for ABSTRACT-Files. Please note that the specified ROBOT program is only read, but not started or executed.

  Example:

```
TF_LOAD  r3demo  mantec_R3 ;
```

- **TF_RUN <teach_filename> <Robot_id> [exec_wait]**

  starts a ROBOT program for execution which was loaded before with TF_LOAD. The optional parameter **exec_wait** is causing a delay of program execution until another program for the specified ROBOT has stopped its execution.

  The execution of the SCRIPT-File is immediately continued since KISMET does not suspend reading the SCRIPT-File until a currently running ROBOT program has stopped.

  Example 1:

```
TF_RUN  r3demo  mantec_R3;
TF_RUN  r3demo2  mantec_R3a;
```

  The two ROBOT programs 'r3demo' and 'r3demo2' are started simultaneously and executed in parallel. Any synchronization between different ROBOTs is not provided by the SCRIPT control-mechanism.

  However, the current implementation level for simulation of IRDATA programs within KISMET allows the synchronization of concurrently running ROBOT programs via IRDATA statements from inside a ROBOT program (use of IRDATA-SEMAPHORES).

Example 2:

```
TF_RUN hs1 ZELLENKRAN;
TF_RUN hs2 ZELLENKRAN exec_wait;
TF_RUN hs3 ZELLENKRAN exec_wait;
```

In the second example, the three ROBOT programs 'hs1..hs3' are activated for execution together, but only 'hs1' will start execution immediately. Execution of 'hs2' is suspended until 'hs1' has stopped, whereas 'hs3' will start after 'hs2' has stopped etc. .

- **TF_DELETE <teach_filename> <Robot_id>**

  deletes the ROBOT program with the name <teach_filename> from the virtual controller of the ROBOT with the name <Robot_id>.

  Example:

  ```
  TF_DELETE  ird_getschr  MANTEC_RM3 ;
  ```

- **LOAD_VIEWS <view_filename>**

  loads a previously in KISMET generated binary view-file. The default extension for view-files is '.vws'. View-Files are stored and searched for in the directory *'$kis_home/scripts'*.

  Example:

  ```
  LOAD_VIEWS view1.vws;
  ```

- **EXEC_SCRIPT <script_file>**

  call and execution of another 'SCRIPT-File' implemented as a SCRIPT-command. This mechanism is similar to calling a subroutine in a high-level programming language.

  The parameter <script_name> may not be the same as the filename of the calling SCRIPT-File. The reason is obvious.

  Example:

  ```
  EXEC_SCRIPT trs_demo.spt ;
  ```

- **DRAW_SCENE**

  the command causes recalculation of the kinematics and rendering of the simulation scenario. This command should be used as a termination command of a sequence of 'FRAME_TO_WFRM'-, 'LOAD_VIEWS', 'ACTIVATE' or 'CONNECT' commands. The commands mentioned do not trigger re-rendering of the model.

  Example:

  ```
  DRAW_SCENE ;
  ```

- **CONNECT <frame_2> TO <frame_1>**

  this command is used to attach kinematically <frame_1> with <frame_2>. The FRAME or

JOINT <frame_1> becomes the kinematic predecessor of <frame_2>. The relative transformation matrix between <frame1> and <frame2> is calculated such, that the relative position between the two COMPONENTS is kept constant subsequently.

This operation can be used for ROBOT-tool exchange or for gripping operations of handled parts. Since the relative position between the 'gripping' frame and the 'gripped' frame is kept, this function is well suited for the simulation of parts and components in any gripping-position. For this reason the ROBOT wrist should be in the correct coupling position in the moment of a tool exchange.

Example:

```
CONNECT  Testlab_1.FL_Klein_02.Flansch_Typ1  TO  Mantec_1.HAND ;
```

- **FRAME_TO_WFRM <frame_2> TO <frame_1> <WFRM_name1>**

  connects the component <frame_1> to <frame_2> at the predefined relative assembly position <WFRM_name1>. The relative transformation between <frame_1> and <frame_2> is defined by the WORKFRAME with the ID <WFRM_name1>.

  This operation can be used - similar to the 'CONNECT'-command - for a ROBOT-tool exchange or for positioning of parts and components. If the components to be coupled are not in the correct position relative to eachother, the component to be connected will "jump" to the defined coupling position.

  Example:

```
FRAME_TO_WFRM Toolbox.Tool1 TO  Mantec_1.HAND Tool1_WFRM ;
```

- **POS_ROB <Robot_id> <mode> <position_data>**

  incremental or absolute positioning of the ROBOT with the name <Robot_id>. The parameter <mode> defines the mode of positioning. Following parameters for **<mode>** are available:

  JOINT_ABS

  absolute positioning in *joint coordinate space*. The number of joint-variable parameters must not exceed the number of DOFs which are defined for the ROBOT model <Robot_id>.

  SYNOPSIS

  JOINT_ABS <j1> <j2> ... <jn>

  JOINT_INC

  incremental positioning in *joint coordinate space*. The number of joint-variable parameters must not exceed the number of DOFs which are defined for the ROBOT model <Robot_id>.

  SYNOPSIS

  JOINT_INC <j1> <j2> ... <jn>

  FRAME_ABS

  absolute positioning in *cartesian space*. The parameters are given in RPY-coordinates,

relative to the ROBOT ZP-FRAME.

SYNOPSIS

FRAME_ABS <x> <y> <z> <r> <p> <y>

FRAME_INC

incremental positioning in *cartesian space*. The parameters are given in RPY-coordinates, relative to the ROBOT ZP-FRAME.

SYNOPSIS

FRAME_INC [X <x>, Y <y>, Z <z>, A <r>, B <p>, C <y>]

The incremental positioning parameters are optional, at least one parameter must be given.

TOOL_INC

incremental positioning in *cartesian tool space*. The parameters are given in RPY-coordinates, relative to the ROBOT TCP-FRAME.

SYNOPSIS

TOOL_INC [X <x>, Y <y>, Z <z>, A <r>, B <p>, C <y>]

The incremental positioning parameters are optional, at least one parameter must be given.

SCREEN_INC

incremental positioning in *screen space*. The parameters are given in RPY-coordinates, relative to the display coordinate system, i.e. x-axis "right/left", y-axis "up/down", z-axis "in/out".

SYNOPSIS

SCREEN_INC [X <x>, Y <y>, Z <z>, A <r>, B <p>, C <y>]

The incremental positioning parameters are optional, at least one parameter must be given.

Examples:

```
POS_ROB MANTEC_RM3 JOINT_ABS 27.5 -17.38 -11.02 37.338 0.9 -12.271 ;
POS_ROB MANTEC_RM3 JOINT_INC -0.1 0 0 0.85 0 -0.3 ;
POS_ROB MANTEC_RM3 FRAME_ABS 1000.0 0 0 -90.0 0 -90.0;
POS_ROB MANTEC_RM3 FRAME_INC X -0.1 Y 0.5 A -0.01;
POS_ROB MANTEC_RM3 TOOL_INC Z 0.182 B 0.001 ;
```

- **SET_FRAME <frame_node> RENDER <draw_parameter>**

  sets for the geometry connected to the specified FRAME a new rendering mode. The FRAME is identified by its name <frame_node>.

  Possible values for <draw_parameter> are: 2 : rendering as wireframe-model 3 : "Flat-Shaded" rendering 4 : "GORAUD-Shaded" rendering 20 : Transparent rendering

Example:

```
SET_FRAME Testlab_1.FL_Klein.Flansch_Typ1 RENDER 2 ;
```

The specified component will be subsequently rendered as wireframe model.

- **SET_FRAME <frame_node> PLACEMENT <place_parameter>**

  sets for the by the identifier <frame_node> specified coordinate system (FRAME) a new relative transformation matrix. The transformation parameters <place_parameter> are to be listed in the order: <wx>,<wy>,<wz>,<dx>,<dy>,<dz> - the three rotational parameters "wx,wy,wz" are to be given in 'degrees', the translational parameters "dx,dy,dz" in 'mm'. All kinematic successors to the repositioned FRAME will be repositioned accordingly.

  This SCRIPT-command allows mainly the possibility of repositioning components in the workcell. The transformation parameters could be delivered "On-Line" from a 3D "measuring system". The coupling to KISMET could be realized through the 'message'-interface.

  Example:

```
SET_FRAME Testlab_1.FL_Klein.Flansch_Typ1 PLACEMENT
          -88.5 16.32851 -180.0 -11920.0 2517.3 912.7 ;
```

- **SET_ABSTRACT <frame_node> RENDER <draw_parameter>**

  sets for the FRAME identified by <frame_node> **and all higher detail levels**, i.e. for all geometries defined in the connected ABSTRACT-Files of type '.mpc', the specified rendering mode. The possible values for <draw_parameter> are defined accordingly to the command 'SET_FRAME'.

- **ROBOT <Robot_id> [online | offline]**

  is used to switch **monitoring-mode** on/off for the ROBOT model as specified by the parameter <Robot_id>. In monitoring mode an interface channel is opened between KISMET and the robot controller. The "real" robot joint variables are sent to KISMET through this channel by the controller. The sensor data received through this channel are used by KISMET to render a synthetic view of the workcell, showing the current position of the robot.

  Example:

```
ROBOT MANTEC_R3 online ;
```

- **ROBOT <Robot_id> [send_on | send_off]**

  starts/stops transmission of robot-parameter messages by KISMET through the "SENSOR" message-channel (key: 1990, type: 5). This message output type is used to monitor KISMET robot simulation by another UNIX-process.

  Example:

```
ROBOT MANTEC_R3 send_off ;
```

- **SET_CAMPAR <CAM_id> <view_dist> <fovy_angle>**

  sets the viewing parameters for the simulated CAMERA with WFRM-name <CAM_id>. The parameters <view_dist> and <fovy_angle) define the focal distance (defined in 'meters'), and the fovy angle (the opening angle of the viewing pyramid, defined in 'degrees'), respectively.

  The function is useful for monitoring of "real" cameras in the workcell. KISMET is used here to display the viewing pyramids of the CAMERAs in the scene. If the cameras are actuated, then the kinematical structure is to be defined in KISMET as a ROBOT.

  Example:

  ```
  SET_CAMPAR CAMERA_5 4.85 15.71 ;
  ```

- **SET_USSPAR active <USS_id> [0 | 1]**

  this function starts/stops ultrasonic sensor (USS) simulation for the sensor identified by its WFRM-name <USS_id>. The value "1" is used to switch the sensor on, "0" to switch off, respectively. If the USS is switched on, KISMET calculates in each cycle a simulated distance value (to the next obstacle) and sends the result through the message output (key: 1990, type: 5).

  Example:

  ```
  SET_USSPAR active USS_1B 0 ;
  ```

- **SET_USSPAR sendmode [0 | 1]**

  this command is used to configure the USS message output mode. When the value "0" (the default) is used, for every sensor one message is sent by KISMET. If the mode is set to "1", an array of range values is sent for all sensors in one message.

  Example:

  ```
  SET_USSPAR sendmode 1 ;
  ```

- **SET_USSPAR range <USS_id> <USS_range>**

  this command is used to set the range of the USS with WFRM_name <USS_id>. The range value <USS_range> (the positive distance in front of the USS where obstacles are detected) is defined in 'mm'.

  Example:

  ```
  SET_USSPAR range USS_5 3500.0 ;
  ```

- **TAG_DRAWLEN <len_value>**

  specifies the length in 'mm' for drawing of workframe coordinate systems. The default value in KISMET is 250mm, which is suitable for medium sized workcells.

  Example:

  ```
  TAG_DRAWLEN 50.0 ;
  ```

- **CAM_TRACKING <cam_id> [on | off] (<frame_id>)**

  is used to switch the CAMERA with WFRM-name <CAM_id> into **tracking mode** and off again. The parameter <frame_id> defines the target frame in the model to track. It is to be defined only if tracking is switched **on**.

  Example:

  ```
  CAM_TRACKING UMG_Kamera_1 on Gantry.J_2.Hand ;
  ```

- **CAM_VIEW_PYRAMID <cam_id> [on | off]**

  this command is used to switch the display of the viewing pyramid on/off for the CAMERA identified by <cam_id>.

  Example:

  ```
  CAM_VIEW_PYRAMID UMG_Kamera_1 on ;
  ```

- **CAM_ONLINE <cam_id> [on | off]**

  is used to switch the "online" mode for the specified CAMERA on/off. In "online" mode, KISMET sends the current CAMERA parameters via message IPC to the UNIX-process which communicates with the application specific camera control system.

  Example:

  ```
  CAM_ONLINE UMG_Kamera_1 on ;
  ```

1.2.2 SCRIPT-Commands for the Definition of ROBOT-Parameters

The following SCRIPT-commands serve for specification of ROBOT parameters and features. The parameters should be set accordingly to the parameters as programmed in the real robot controller. It is recommended to create for each robot type in the library a separate SCRIPT-File to define its kinematical parameters and control behavior. All parameters used by KISMET for ROBOT movements and program simulations are initially set to reasonable default values. The SCRIPT-File should set these values according to the start state of the real robot controller to have reasonable parameters for 'Off-Line' programming and simulation.

Note: Since the commands listed in the following chapters all refer to the *active ROBOT*, the SCRIPT-command *ROBACT* should be executed before any of these commands.

- **rob_cart_lin_vel <lin_vel>**

  Defines the maximum velocity for cartesian linear movements in the unit 'mm/sec'.

  Example:

  ```
  rob_cart_lin_vel 1200.0;
  ```

- **rob_cart_lin_acc <lin_acc>**

  Defines the maximum acceleration for cartesian linear movements in the unit 'mm/sec^2'. This parameter defines together with <lin_vel> the duration for the acceleration period for

cartesian movements. In the chosen example, the acceleration period would become together with <lin_vel> 250ms.

Example:

```
rob_cart_lin_acc 4800.0;
```

- **rob_cart_rot_vel <ang_vel>**

  Defines the maximum possible velocity for tool orientation changes during cartesian movements. The parameter is defined in the unit: 'degrees/sec'.

  Example:

  ```
  rob_cart_rot_vel 360.0;
  ```

- **rob_cart_rot_acc <lin_acc>**

  Defines the maximum angular acceleration for tool orientation rotations during cartesian linear movements. The unit used is for this parameter is 'degrees/sec^2'.

  Example:

  ```
  rob_cart_rot_acc 720.0;
  ```

- **rob_cart_vel_scal <lin_vel>**

  Defines the velocity scaling-factor for cartesian linear movements as a certain percentage of the maximum velocity (see 'rob_cart_lin_vel'). The possible range for <lin_vel> is [0.01..1.0], corresponding to 1% to 100%.

  In the example, the current cartesian velocity is set to 30% of the maximum value. The scaling for the linear velocity is also scaling the current parameter for orientation changes by the same amount.

  Example:

  ```
  rob_cart_vel_scal 0.3;
  ```

- **rob_cart_acc_scal <lin_acc>**

  Defines the acceleration scaling-factor for cartesian linear movements as a certain percentage of the maximum acceleration (see 'rob_cart_lin_acc'). The possible range for <lin_acc> is [0.01..1.0], corresponding to 1% to 100%. In the example, the current cartesian acceleration is set to 65% of the maximum value.

  Example:

  ```
  rob_cart_acc_scal 0.65;
  ```

- **rob_cart_fly_radius <fly_type> <fly_parameter>**

  Defines the behavior during robot program execution for flypoints for linear (cartesian) motion. The parameter <fly_type> (integer) defines the calculation mode during flypoint approach, depending on this type the <fly_parameter> (float) defines the quantitative

parameter. The modes are available as follows:

fly_type	flypoint behavior for program simulation
0	If the linear velocity has dropped below the percentage as defined by <fly_parameter> in the range [0 .. 1], a flypoint is detected and the next (linear) motion command starts execution.
1	If the position of the TCP is within the distance from the target position as defined by <fly_parameter> (in 'mm'), the next (linear) motion command is started.
2	Similar as mode 1, but the angular distance is defined (in 'degree'). The parameter defines the difference angle from the target orientation.

Example:

```
rob_cart_fly_radius 1 5.0 ;
```

- **rob_program_delays <delay>**

  Defines a delay in seconds for robot program execution, i.e. a simulated "delay" for program startup (after TF_RUN). This is useful for execution time simulation of program sequences. The delays may be caused in real life by program downloads to the controlles or other causes. The delay is defined in seconds.

- **rob_jnt_max_range <njoints>[max_pos]>**

  Defines the upper mechanical limit <max_pos> of the moving range for the different robot axes. The parameters are to be listed in the same order as the joints are defined in the corresponding '.mpc'-File. The parameters are defined in 'mm' for prismatic joints, and 'degrees' for rotational joints, respectively.

  Example:

```
rob_jnt_max_range 162.5 110.0 145.0 187.5 115.5 270.0;
```

- **rob_jnt_min_range <njoints>[min_pos]>**

  Defines the lower mechanical limit <min_pos> of the moving range for the different robot axes. The whole mechanical range of possible movements is not necessarily symmetric to the zero- or to the reference position.

  Example:

```
rob_jnt_min_range -162.5 -110.0 -145.0 -187.5 -115.5 -270.0;
```

- **rob_jnt_start_pos <njoints>[start_pos]>**

  Defines the initial position of the ROBOT at the beginning of the simulation.

  Example:

```
rob_jnt_start_pos 0.0 50.0 -110.0 0.0 -150.0 -180.0;
```

- **rob_jnt_ref_pos <njoints>[ref_pos]>**

  Defines a specific reference position of the ROBOT. This position is usually the reference position of the robot controller. It is often to be defined if incremental position sensors are used by the robot system.

  Example:

```
rob_jnt_ref_pos 17.35 0 -12.79307 1.507E-03 0.0 15.2003;
```

- **rob_jnt_max_vel <njoints>[max_vel]>**

  Defines the absolute value for the maximum joint velocities for a single robot axis. The velocity is defined here in the units 'mm/sec' (prismatic joints), and 'rad/sec' (rotational joints), respectively.

  Example for a 6-DOF industrial robot:

```
rob_jnt_max_vel 1.919862 1.483553 5.235988
                3.14159 4.014257 3.926991 ;
```

- **rob_jnt_min_vel <njoints>[min_vel]>**

  Some robot systems also have a minimum joint velocity, typically when stepper motors are used as actuators. This minimum velocity can influence the behavior of the robot for cartesian linear movements.

  This minimum velocity for single robot joints is defined for the simulation via the command <rob_jnt_min_vel>. Again the units 'mm/sec' and 'rad/sec' are used.

  Example for a 3-axis robot:

```
rob_jnt_min_vel 8.5 3.14159E-04 4.014257E-04 ;
```

- **rob_jnt_act_vel <njoints>[act_vel]>**

  Defines the current value for single robot axis velocities after the controller is switched on. This value (the initial parameter) is used for simulation of robot programs if inside the program the current single-axis velocity is not set before the first movement command is executed. The units used are similar to <rob_jnt_max_vel>.

  Example for a 3-axis robot:

```
rob_jnt_act_vel 1300.0 3.14159 1.5707963;
```

- **rob_jnt_max_acc <njoints>[max_acc]>**

  Defines the current value for single-axis accelerations. The acceleration is to be defined in the units 'mm/sec^2' (prismatic joints), and 'rad/sec^2' (rotational joints), respectively.

  Example for a 6-DOF industrial robot:

```
rob_jnt_max_acc 1.919862 1.483553 5.235988
                3.14159 4.014257 3.926991;
```

- **rob_jnt_act_acc <njoints>[act_acc]>**

  Defines the current value for single robot axis accelerations after the controller is switched on. This value (the initial parameter) is used for simulation of robot programs if inside the program the current single-axis acceleration is not set before the first movement command is executed. The parameter is defined in the units defined for <rob_jnt_max_acc>.

  In the following example the acceleration is set to 100% of the defined maximum value.

  Example for a 6-axis robot:

```
rob_jnt_act_acc 1.919862 1.483553 5.235988
                3.14159 4.014257 3.926991 ;
```

- **rob_PTPmode [sync|async]**

  defines the behavior of the robot controller for PTP-movements. KISMET is able to simulate synchronous (parameter: sync) or asynchronous (parameter: async) PTP-movements. The default is **sync**.

  Example:

```
rob_PTPmode sync;
```

- **rob_por_def <X, Y, Z, P, R, Y>**

  Redefines the current offset for the robots "point-of-reference" (POR). Linear offsets are defined in 'mm', the angular offsets are given in 'degrees'.

  Example:

```
rob_por_def 418.0 0.0 –78.0 90.0 –45.0 0.0
```

- **rob_tcp_def <X, Y, Z, P, R, Y>**

  Redefines the current offset for the robots "tool-center-point" (TCP). Linear offsets are defined in 'mm', the angular offsets are given in 'degrees'.

  Example:

```
rob_tcp_def 83.0 -170.0 0.0 -90.0 0.0 -30.0
```

- **rob_vel_profile [sine|ramp]**

  defines the simulation mode for motion simulation during robot program simulation. Either sinusoidal motion profiles or ramp profiles are possible. The sinusoidal profile (the default) is closer to the "real" motion profiles. For ramp profiles, a "jump" is assumed in the acceleration profile.

  Example:

```
rob_vel_profile ramp;
```

- **rob_inverse_class <solution_hint>**

defines the class (group) of the actual ROBOT for the general inverse kinematics solution. The parameter <solution_hint> gives the textual identifier for the optimisation to be chosen by the algorithm. For mechanisms with more than 6 DOF's / joints, this command defines the optimisation method to be chosen during inverse kinematics movements. For mechanism with less than 6 DOF's / joints, the parameter <solution_hint> defines which TCP axis shall be aligned regarding *position* and/or *orientation* with the target world frame.

Note: The KISMET software is unfortunately not "clever enough" to detect the kinematic structure automatically in every possible kinematic definition file for chosing an optimum solution ! So we leave this to the experienced kinematics modeller.

<solution_hint>	**Optimisation method used by the algorithm**
`align_FRAME`	align *position* and *orientation* of the TCP with the target frame
`align_POS`	align *position* only

`align_X_AXIS`
`align_Y_AXIS`
`align_Z_AXIS`
`align_XY_AXIS`
`align_YZ_AXIS`
`align_XZ_AXIS`
	align the respective *positions*. Try to align all *orientations*. This is for robots with PRRR or PPRRR structure

`align_N_AXIS`
`align_S_AXIS`
`align_A_AXIS`
	align all positions. Align TCP *n/s/a* orientations with the cartesian target frame, the other *orientation* vectors may take any orientation. This is for robots with PPPRR or RRRRR structure (or other 5 DOF robots)

`align_XY_N_AXIS`
`align_XY_S_AXIS`
`align_XY_A_AXIS`
`align_XZ_N_AXIS`
`align_XZ_S_AXIS`
`align_XZ_A_AXIS`
`align_YZ_N_AXIS`
`align_YZ_S_AXIS`
`align_YZ_A_AXIS`
	align respective *positions* and *orientation* vectors as indicated by the name. This mode is intended for robots which can only move in one plane (like the boom-type robots) and can orient their end-effector (TCP) with limitations

`EDITH`
`BOOM_3`
`BOOM_4`
`BOOM_5`
`BOOM_6`
	Articulated-Boom type kinematical structures which can move in the horizontal plane but with redundant kinematics (snake like). This is for some specific KISMET applications.

Example:

```
rob_inverse_class align_POS ;
```

- **ARC [on | off]**

  is used to switch the graphical "sparc-effect" for the active ROBOT at the location of the TCP either ON or OFF. This SCRIPT command is usually issued from inside an IRDATA robot program for the simulation of spot- or arc-welding operations.

  Example:

```
ARC on ;
```

1.2.3 SCRIPT-Commands for Animation

- **STDP <display_id> [FULL | FULL_OFF]**

  STDP (Set_Display) is used to switch the KISMET display into the specified mode. The parameter "FULL" or "FULL_OFF" defines "fullscreen" mode or normal display mode for the chosen view layout. The parameter <display_id> corresponds to the display modes in the "Display" menu:

  1 Single perspective
  2 Top + side view (orthogonal projection)
  3 Single top-view (orthogonal)
  4 Single side-view (orthogonal)
  5 Side view perspective
  6 Birdseye view

  Example:

```
STDP 1 FULL;
```

- **STVW <view_id> [VP <x> <y> <z>] [LP <x> <y> <z>]**
  [TW <twist_angle>] [VA <view_angle>]
  [FCP <front_cp_dist>] [BCP <back_cp_dist>]
  [STEPS <n> | TIME <sec>]

  STVW (Set_View) is used to control via the SCRIPT-interface the parameters of the specified view <view_id>. For the definition of <view_id> see the command 'COVW'. In the parameter list the following view features can be changed:

  VP changes the *viewpoint* parameters

  LP is used to change the *lookpoint* (point-of-interest)

  VA sets the *viewing angle* (Zoom). in vertical direction. The angle is defined in 'degrees'

  TW changes the *twist angle* of the view, i.e. the rotation around the line-of-sight. The angle is defined in 'degrees'

  FCP distance to the *front clipping plane*

  BCP distance to the *back clipping plane*

  STEPS the view sequence is carried out in <n> steps (updates of the scene)

  TIME the view sequence is carried out in the defined time interval. The interval is defined in seconds.

Examples:

```
STVW 0 VP 1238 1511 -662.28 VA 22.5 STEPS 35;
STVW 0 VP 519.7 -539 -850 LP 2183 9570 3391 TIME 15.0;
```

- **COVW <view_id> <cam_id>**

  COVW (Connect_view) binds the defined view to a specific model CAMERA. If that CAMERA is in tracking mode or if it executes a ROBOT-program, the viewing parameters of that camera are copied into the defined view. The parameter <view_id> corresponds to the view array as defined in the layout file (.lay). The standard setting is:

  0 Single perspective view (Display mode 1)
  1 Top view in display mode "top + side"
  2 Side view in display mode "top + side"
  3 Single top-view
  4 Single side-view
  5 Side view perspective
  6 Birdseye view

  Example:

  ```
  COVW 0 UMG_Kamera_1;
  ```

- **SET_MODE <mode> [on | off]**

  The SET_MODE command is used to control several display and viewing parameters via the SCRIPT interface. Possible identifiers for <mode> are:

WFRM_display	Is used to activate or deactivate the display of workframes in the simulation scene
SMEAR	Is used to control the 'smear' mode, i.e. the drawing of a trace curve of the active robots TCP position and / or orientation
BFRM	Backface removal is switched on or off (OpenGL mode)
CPU_BFRM	Backface removal (calculated by the CPU) is switched on or off
CAM_MAINVIEW	connects the current camera view (a camera workframe must be defined in the scene model) to the main display area, i.e. a large camera view occupying the whole display area is achieved. The 'byrdseye' view is used to for viewing parameter management.
LGT_TWOSIDE	Twoside lighting mode is switched on or off (OpenGL feature).
TCP_DRAW	Display of the active robots TCP frame is switched on / off.
ZPF_DRAW	Display of the active robots ZPF frame is switched on / off.
FLDR	Automatic "floor" display is switched on or off.
GEO_VIS_TEST	Geometry visibility test on or off.
SHAD	(fake) shadow display on or off. This mode requires the activation of the automatic "floor" (see above).
BEEP	Activates or deactivates the "beep" sound with message display by the KISMET kernel.
TEXTURES	Hardware texture mapping display mode on/off.
ANTIALIAS	Scene antialiasing mode on/off.

- **BFRM [on | off]**

  is used to switch **backface removal** on/off.

- **FLDR [on | off]**

  is used to switch **floor drawing** on/off.

- **SHAD [on | off]**

  activates the calculation and display of **real-time shadows** (fake shadows).

- **WAIT**

  the execution of the current SCRIPT-file is suspended until all active ROBOT programs and viewing sequences are finished.

1.2.4 SCRIPT-Commands for Multibody-Dynamics Simulation

- **SET_FORCE_ARROW <int_1> <int_2> <int_3>**

  Defines the parameter for the **Force Display** option (see "The KISMET Guide to DYNAMICS").

  <int-1> [0 | 1 | 2 | 3 | 4]

  Parameters for the force arrows.

  0 = Draw only the X-componet
  0 = Draw only the Y-componet
  0 = Draw only the Z-componet

  <int_2> [0 | 1 | 2 | 3 | 4]

  Parameters for the force arrows. (see <int_1>)

  <int_3> [0 | 1 | 2 | 3 | 4]

  Parameters for the TCP-force arrows. (see <int_1>)

- **SET_TORQUE_LIMIT <max_force> <max_torque>**

  Defines limits for forces and torques during interaction with elastodynamical objects (see **TCP-Ext.Force** command. If the forces/torques are violating the limits a warning will appear on the screen if

- the limits are ¹ *0*
- and the force/torque values are send to via Shared Memory (key 1313) to another process

- **SET_EXT_TCP_TEST [on | off] <value_1> <value_2> <value_3> <value_4>**

  Enables or disables the test mode for the **TCP-Ext.Force** command. The specified values (<value_1>, ... ,<value_4>) will be send to Shared Memory (key 1313) if test mode is enabled.

- **SET_FRC_ARR_SPECIAL [on | off]**

  Correspond to the SHOW ONLY MOTOR FORCES/TORQUES button in the Set FORCE Display command in the DYNAMICS Option Panel. Only the component of the force/torque vector that correspond to the motor torques/forces (Z-component of the torque vector for rotational DOFs and Z-component of the force vector for translational DOFs) will be visualized.

- **MASTER_SLAVE [on | off] <robot_id> <number_of_slaves>**
  < <robot_id>,<number_of_dof>
  < <name_of_master_joint> <name_of_slave_joint>
  > * number_of_dof
  > * number_of_slaves

  Enables/disables the Master-Slave option:

  The first robot_id defines the master mechanism the others up to 5 the slaves mechanisms. The master DOFs will be connected to the specified slave DOFs. For identification the joint names defined in the mpc-files are used.

1.2.5 SCRIPT-Commands related to elastodynamical Objects

These SCRIPT commands are used to modify elastodynamic or geometric data of deformable objects in KISMET. Additionally, they are used to set specific rendering parameters for these objects, global interconnections (hyper data) between deformable parts and for the definition of parameters used for simulation of interactions between mechanisms (e.g. surgical instruments) and deformable objects (e.g. tissue and organ models in a surgery simulation). Another group of commands is used to control simulation effects.

- **eladyn_geo_def [<geo_name> | ALL] <flags>**
  { <value1> <value2> <value3> <value4> }

This command defines object attributes for one or all elastodynamic objects in a simulation model.

<geo_name> [*string*]
The filename of an elastodynamic object in the KISMET specific format ('*.mpo'). The string 'ALL' denotes all deformable objects in the simulation model.

<flags> [*unsigned*]
Attribute field, defines object specific behavior, calculation and visualization parameters
⇒ see table A.1

<value1 .. 4> [*float*]
simulation specific modification of parameters (currently not used)

- **eladyn_geo_def [<geo_name> | ALL] <token> [on | off]**

The command in this syntax defines a single object attribute for one or all elastodynamic objects. The value 'on' activates the option, the value 'off' deactivates the option.

<geo_name> [*string*]
The filename of an elastodynamic object in the KISMET specific format ('*.mpo'). The string 'ALL' denotes all deformable objects in the simulation model.

<token> [*string*]
Keyword for the object attribute to modify.
⇒ see table A.1

- **eladyn_hyper_name <hyper_file>**

Defines the filename of the ***ELADYN_HYPER-file***. This file defines the global interconnections between deformable objects.

<hyper_file> [*string*]
the filename of the ELADYN_HYPER-file (located in the KISMET model directory *'$kis_home/envlib'*)

- **eladyn_show_def [0 | 1]**

SCRIPT command to define visualization attributes for elements of the elastodynamic nodal network, e.g. the virtual mass nodes and the interconnecting spring elements (0: visualization off, 1: visualization on). Mass nodes are rendered as tetrahedron elements, which are rendered in different colors, depending on their attributes (inner and outer nodes) and their interaction state. The connecting elements (virtual springs) are rendered as lines between the mass nodes. Additionally, force vectors and the orientation of the central node are visualized.

- **eladyn_model_reset**

The command resets the structure of all elastodynamical objects and of the global interconnections to their initial state in the simulation.

- **eladyn_plaus_param <token> <flags> <val>**

This command is used to modify some parameters in the plausibility test which are used in the interaction modules of the simulation. The parameters can be adapted to the specific requirements of a simulation model during development.

<token> [*string*]
Keyword for the definition of the level and location for the plausibility test.
⇒ see table A.5

<flags> [*unsigned*]
defines criteria for the plausibility test, i.e. the type of processing
⇒ see table A.4

<val> [*float*]
parameter limit value for the plausibility test

- **EFFECTS <mode> [ON | OFF]**

The EFFECTS command is used to control several aspects of simulation only possible with elastodynamically modelled objects. The available set of subcommands defined by the <mode> identifier, is mainly used in surgery simulation, but other applications could be possible. Possible identifiers for <mode> are:

PULSE	Is used to activate or deactivate the simulation of the "pulse" (hemodynamic effect) in arterial vessels and bleeding tests. A wave of forces (the cyclic "blood pressure") is applied to all nodes in the tissue model of the "artery".
BLEEDING	Activates or deactivates bleeding simulation. If a blood vessel is cut, a particle system simulation is used to simulate the arterial bleeding.
POOL_RIPPLES	Enables or disables the simulation of water ripples on the pool surface during irrigation and dipping/moving surgical instruments in the fluid pool.

- **EFFECTS <mode>**

Another set of EFFECTS commands is used without the [ON | OFF] parameter. Possible

identifiers for <mode> are:

RESET_POOL Reset the color, transparency and the height of the accumulated fluid after irrigation and bleeding inside the virtual body.

CLEAN_ENDOSCOPIC_LENSE
Reset the transparency of the endoscopic camera lense, which has been steamed up after some period of coagulation.

1.2.6 Deformable Object Simulation Control and I/O-channels

These SCRIPT commands are used for the application specific configuration of the KISMET kernel for interactive simulation of elastodynamically deformable objects. There are commands available for the definition of input/output data-channels used to connect various input devices to the KISMET kernel, e.g. binary switches or haptic devices like the PHANTOM device (Sensable Technologies Inc.) or the IMPULSE ENGINE device (Immersion Corp.). Other commands are used to control and/or optimize elastodynamics simulation.

- **eladyn_calc [on | off]**

Toggles the elastodynamics calculation and the interaction processing. It is important to switch the elastodynamics off (command: eladyn_calc off) before the simulation model is modified (by example with a RESET function). Otherwise the simulation data-model may become inconsistent and subsequent system crashes could result. Note that the elastodynamics is calculated as a parallel process on multiprocessor workstations.

- **rob_ela_interact <int_jnt> <test_function> <testval1> <testval2> <inst_type>**

Defines the behaviour of the ACTIVE ROBOT (see SCRIPT command 'ROBACT') when used for interaction with elastodynamics and surgery simulation (gripping, cutting, coagulation, suction, clipping etc.). Especially the effector interaction is defined. If the defined condition is valid, the specific function is performed with the elastodynamic object in contact with the instrument.

<int_jnt> [*unsigned*]
defines the effector degree-of-freedom (DOF), which is responsible for the specific action of the instrument, i.e. the DOF's index from the ROBOT definition file (see specification of '.dof' files).

<test_function> [*unsigned*]
defines a test function which triggers the action of the instrument effector. The current value of the effector DOF (defined by <int_jnt>) is <jnt_val>. The user defined constants <testval1> and <testval2> define the limits for triggering the effector action. Avalaible test functions are:

0: no test function (interaction off)
1: jnt_val ≥ testval1
2: jnt_val < testval1
3: testval1 ≤ jnt_val ≤ testval2
4: (jnt_val < testval1) OR (jnt_val > testval2)

<inst_type> [*unsigned*]
Defines the effector type of the active ROBOT (instrument), and specifically the action associated with the effector. Available instrument types are:
0: Surgical Gripper
1: Clip-Applicator
2: Scissors (Action: cutting)
3: Coagulation hook
4: Coagulation foreceps

- **ela_clip_def <clip_file><trans_rx><trans_ry> <trans_rz><trans_tx><trans_ty><trans_tz>**

Defines the relative position of a clip with respect to the end-effector for the instrument *clip-applicator*. The geometry ('.mpo' filename) and its relative position transformation to the clip instruments end-effector are definable.

<clip_file> [*string*]
defines the filename for the geometry of the clip ('*.mpo')

<trans_rx> [*float*]
<trans_ry> [*float*]
<trans_rz> [*float*]
Rotational transformation parameters in 'degrees' around the x-, y- and z-axes, respectively.

<trans_tx> [*float*]
<trans_ty> [*float*]
<trans_tz> [*float*]
Translational transformation parameters in 'mm' along the x-, y- and z-axes

- **eladyn_geo_def <geo_name> <flags> geom1> <geom2> <geom3> <geom4>**

For performance optimization, a cylinder test is performed for collision testing between surgical instrument effector geometry and deformable object(s). Note that not the geometry as defined during instrument modeling is tested, but the virtual cylinders around the effector parts, as defined with this SCRIPT command for the *active ROBOT*.

<geom1> [*float*]
Defines the shaft diameter of the simulated instrument effector [mm]

<geom2> [*float*]
average distance to the outside effector surface from the virtual end-effector axis. Given in [mm], measured in end-effector motion direction.

- **rob_ela_shm <send_type> <re_shmkey> <re_buflen>**

Defines a shared memory data channel for the transmission of force-data and joint-torques to the control process (program, thread) of a haptic input device used together with the KISMET software in elastodynamic simulations. All forces to be exported are calculated relative to the simulated instruments end-effector. The forces are transmitted by KISMET in the unit 'Newton', the force is written into the shared memory data buffer using 'float' (32-Bit) data-size.

<send_type> [*unsigned*]
defines the type of shared memory data exchange (the protocol). Currently not used.

<re_shmkey> [*unsigned*]
defines the *shared memory key*. The *key* is used by all processes accessing the shared memory (read or write).

<re_buflen> [*unsigned*]
size of shared memory in bytes

- **bin_ela_shm <send_type> <be_shmkey> <be_buflen>**

The command defines a common memory area (shared memory) for the exchange binary data (digital I/O) of elastodynamical simulation data. The command is used for application specific configuration of the KISMET kernel. In the MIC-trainer application, this command is used to define the input channel for foot switches. The binary input buffer is evaluated bitwise. If any bits are changing their state (0 to 1 or 1 to 0), the SCRIPT-files with predefined names:

BININP1_ON.spt	(Bit 0 changes its state from 0 to 1)
BININP1_OFF.spt	(Bit 0 changes its state from 1 to 0)
BININP2_ON.spt	(Bit 1 changes its state from 0 to 1)
BININP2_OFF.spt	(Bit 1 changes its state from 1 to 0)
BININP3_ON.spt	(Bit 2 changes its state from 0 to 1)
BININP3_OFF.spt	(Bit 2 changes its state from 1 to 0)
BININP4_ON.spt	(Bit 3 changes its state from 0 to 1)
BININP4_OFF.spt	(Bit 3 changes its state from 1 to 0)
BININP5_ON.spt	(Bit 4 changes its state from 0 to 1)
BININP5_OFF.spt	(Bit 4 changes its state from 1 to 0)
BININP6_ON.spt	(Bit 5 changes its state from 0 to 1)
BININP6_OFF.spt	(Bit 5 changes its state from 1 to 0)
BININP7_ON.spt	(Bit 6 changes its state from 0 to 1)
BININP7_OFF.spt	(Bit 6 changes its state from 1 to 0)
BININP8_ON.spt	(Bit 7 changes its state from 0 to 1)
BININP8_OFF.spt	(Bit 7 changes its state from 1 to 0)

are executed by the KISMET kernel. The parameters of the SCRIPT command are to be used as:

<send_type> [*unsigned*]
defines the type of shared memory data exchange (the protocol). Currently not used.

<be_shmkey> [*unsigned*]
defines the *shared memory key*. The *key* is used by all processes accessing the shared memory (read or write).

<be_buflen> [*unsigned*]
size of shared memory in bytes

- **BIN_ELA [online | offline]**

Activates (parameter: online) or deactivates (parameter: offline) the processing of binary input data channels as defined with the *bin_ela_shm* SCRIPT command.

- **hot_message <text>**

Alphanumeric display of system or simulation messages on the graphics display (for about 5 seconds). Can be used for the definition of user interaction messages or for the display of simulation results.

<text> [*string[..]*] message text to display

1.2.7 SCRIPT-Commands for 3D-Texture Voxel-Volume Visualization

A set of SCRIPT commands is available in KISMET for 3D-texture based voxel-volume visualization. Of cause, these commands can be effective only if any volume data-sets are loaded into the current simulation scenario.

- **VV_CLIP <Clippingplane> [active | reverse | visible | off]**

KISMET provides 6 clipping planes specific for the display of 3D-texture volumes. The SCRIPT command is used to control these clipping planes.

<Clippingplane> [*string*] values: CLP1, CLP2, CLP3, CLP4, CLP5, CLP6

- **VV_CLIP_POS <Clippingplane> ELE <elevation> AZI <azimuth> DIST <distance>**

Controls the position of a clipping plane. The parameters define a plane equation. The clipping plane is defined with reference to the voxel-volume coordinate system.

<elevation> [*float*]
Defines the elevation angle of the clipping plane in 'degrees'. Range: [-90.0 ... 90.0]

<azimuth> [*float*]
The azimuth angle of the clipping plane in 'degrees'. Range: [-90.0 ... 90.0]

<distance> [*float*]
Defines the shortest distance of the clipping plane to the origin of the reference coordinate system in 'mm'.

- **VV_SET_LUT <LUT_name>**

Activates the 3D-texture color lookup table denoted by the parameter <LUT_name>. Parameters are:

<LUT_name> [*string*] values: MEDICAL_LUT
LIN_LUT
EXP_LUT
RAINBOW_LUT
USER1_LUT
USER2_LUT

- **VV_SET_LUT_PAR MEDICAL_LUT <air> <fat> <tissue> <bone>**

- **VV_SET_LUT_PAR <Lut_name> <opacity> <gain> <level> <window>**

The special lookup table *MEDICAL_LUT* is precalculated for visualization of medical CT-data (X-Ray) according to their grayscale value range. The parameters define the grade of display or intensity for various tissue/material types (air, fat, tissue, bone). The useful value range of the four float parameters is the interval [0.0 ... 1.0].

For all other LUT's, the parameters define the attributes of the table

<opacity>	[*float*]	range: 0.0 .. 3.0
<gain>	[*float*]	range: 0.0 .. 20.0
<level >	[*float*]	range: 1.0 .. 3.0
<window>	[*float*]	range: 0.0 .. 1.57

- **VV_RELOAD_LUTS**

Reloads all 3D-texture lookup tables.

1.3 Example of a SCRIPT Commandfile

```
RBv01r01
#
# SCRIPT_File 'trsys_demo.spt':
#
#   Loads the higher detailing levels for the flanges to be
#   exchange and the IRDATA robot programs for the gantry and
#   the MANUTEC-R3 industrial robot. The views are modified
#   to zoom the handling objects and display them in the centre
#   of the different views. The IRDATA program for the
#   CATROB-gantry which moves to the reference position is
#   started immediately.
#
SWAP_NODE HTL_UMGEBUNG.FL_Klein01;
SWAP_NODE HTL_UMGEBUNG.FL_Klein02;
ROBOT_LANGUAGE irdata;
TF_LOAD ird_trstart TRAEGERSYSTEM;
TF_LOAD ird_711 MANTEC_RM3;
LOAD_VIEWS trs_demo.vws;
TF_RUN ird_trstart TRAEGERSYSTEM;
FRAME_TO_WFRM Toolbox.Tool1 TO  Mantec_1.HAND Tool1_WFRM
```

2 SCRIPT-Command Summary

ACTIVATE <frame_node>

ARC [on | off]

BFRM [on | off]

CAM_ONLINE <cam_id> [on | off]

CAM_TRACKING <cam_id> [on | off] (<frame_id>)

CAM_VIEW_PYRAMID <cam_id> [on | off]

CONNECT <frame_2> TO <frame_1>

COVW <view_id> <cam_id>

DEACTIVATE <frame_node>

DELETE_NODE <frame_node>

DRAW_SCENE

EXEC_SCRIPT <script_file>

FLDR [on | off]

FRAME_TO_WFRM <frame_2> TO <frame_1> <WFRM_name1>

LEVEL_DOWN <frame_node>

LEVEL_UP <frame_node>

LOAD_VIEWS <view_filename>

MASTER_SLAVE [on | off] <robot_id> <number_of_slaves><

<robot_id>,<number_of_dof><<name_of_master_joint> <name_of_slave_joint>

>* number_of_dof> * number_of_slaves

POS_ROB <Robot_id> <mode> <position_data>

rob_cart_acc_scal <lin_acc>

rob_cart_lin_acc <lin_acc>

rob_cart_lin_vel <lin_vel>

rob_cart_rot_acc <lin_acc>

rob_cart_rot_vel <ang_vel>

rob_cart_vel_scal <lin_acc>

rob_inverse_class <class_id>

rob_jnt_act_acc <njoints>[act_acc]>

rob_jnt_act_vel <njoints>[act_vel]>

rob_jnt_max_acc <njoints>[max_acc]>

rob_jnt_max_range <njoints>[max_pos]>

rob_jnt_max_vel <njoints>[max_vel]>

rob_jnt_min_range <njoints>[min_pos]>

rob_jnt_min_vel <njoints>[min_vel]>

rob_jnt_ref_pos <njoints>[ref_pos]>

rob_jnt_start_pos <njoints>[start_pos]>

rob_PTPmode [sync|async]

ROBACT <Robot_id>

ROBOT <Robot_id> [online | offline]

ROBOT <Robot_id> [send_on | send_off]

ROBOT_LANGUAGE <language>

SET_ABSTRACT <frame_node> RENDER <draw_parameter>

SET_CAMPAR <CAM_id> <view_dist> <fovy_angle>

SET_EXT_TCP_TEST [on | off] <value_1> <value_2> <value_3> <value_4>

SET_FORCE_ARROW <int_1> <int_2> <int_3>

SET_FRAME <frame_node> PLACEMENT <place_parameter>

SET_FRAME <frame_node> RENDER <draw_parameter>

SET_FRC_ARR_SPECIAL [on | off]

SET_TORQUE_LIMIT <max_force> <max_torque>

SET_USSPAR active <USS_id> [0 | 1]

SET_USSPAR range <USS_id> <USS_range>

SET_USSPAR sendmode [0 | 1]

SHAD [on | off]

STDP <display_id> [FULL | FULL_OFF]

STVW <view_id> [VP <x> <y> <z>] [LP <x> <y> <z>]
[TW <twist_angle>] [VA <view_angle>]
[FCP <front_cp_dist>] [BCP <back_cp_dist>]
[STEPS <n> | TIME <sec>]

SWAP_NODE <frame_node>

TAG_DRAWLEN <len_value>

TF_DELETE <teach_filename> <Robot_id>

TF_LOAD <teach_filename> <Robot_id>

TF_RUN <teach_filename> <Robot_id> [exec_wait]

WAIT

Index

A

B

C

D

E

F

H

L

M

P

R

S

T

V

W